Preprint
UCRL-JC-140415

An Adaptive Path Planning Algorithm for Cooperating Unmanned Air Vehicles

C.T. Cunningham, R.S. Roberts

This article was submitted to
2001 IEEE International Conference on Robotics and Automation,
Seoul, South Korea, May 21-26, 2001

September 12, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Submitted to the IEEE International Conference on Robotics and Automation, May 21-26, 2001, Seoul, Korea

An Adaptive Path Planning Algorithm for Cooperating Unmanned Air Vehicles

Christopher T. Cunningham
Mail Stop L-181
cunningham2@llnl.gov

Randy S. Roberts *
Mail Stop L-086
roberts38@llnl.gov

Lawrence Livermore National Laboratory
Livermore, CA 94550 USA

Abstract

An adaptive path planning algorithm is presented for cooperating Unmanned Air Vehicles (UAVs) that are used to deploy and operate land-based sensor networks. The algorithm employs a global cost function to generate paths for the UAVs, and adapts the paths to exceptions that might occur. Examples are provided of the paths and adaptation.

1 Introduction

Large networks of land-based sensors are becoming increasingly important for sensing signals produced by natural and man-made phenomena. The deployment and operation of large land-based sensor networks can pose difficult problems, particularly in time critical situations or in rugged terrains. An example of such a scenario is monitoring ground conditions in a forest to model and predict the advance of a forest fire. Autonomous deployment and operation of land-based sensor networks in such a scenario is highly desirable for maximizing sensing efficiency, while minimizing human risk.

An attractive approach to the deployment and operation of a land-based sensor network is to use cooperating Unmanned Air Vehicles (UAVs) to deploy the sensors and then serve as communication hubs for the sensors. In this approach, a group of cooperating UAVs deploy sensors over a region of interest. After the sensors have been deployed, the sensor network is logically partitioned into sub-networks (subnets), with one UAV assigned per subnet. Partitioning the network into subnets allow the UAVs to service sensors in parallel while minimizing interference or duplication of effort. A UAV services sensors in its subnet by flying a route (path) through the subnet, uplinking data collected by the sensors, and forwarding the data to a central point.

In the sensing architecture described above, the UAVs form a loosely cooperative system. Deployment of sensors requires cooperative behavior such as flying in formation, coordinated release of sensors, and so on. After sensor deployment, cooperation is maintained through the exchange of state information between UAVs. State information includes not only the status of UAVs but also the status of sensors in the network. Exchange of state information is necessary so that exceptional events can be detected and properly managed by the UAVs. As an example, consider that a UAV is required to leave the network for replenishment of fuel. The sensors in its subnet still require servicing, so UAVs in neighboring subnets must address this requirement. In order to balance service times for other subnets in the network, the subnet structure of the network adapts to the loss of a UAV.

In this paper, we focus on an important aspect of the architecture, namely the adaptive path planning algorithm. A key requirement of the algorithm is that it adapts the subnets (and paths through the subnets) in response to one of four basic exceptions. The four basic exceptions are: 1) one UAV leaves the network, 2) one UAV enters the network, 3) several sensors leave the network and 4) several sensors enter the network. More complex situations can be derived as combinations of these basic exceptions. In the algorithm presented here, subnet and path adaptation is driven by a global cost function that essentially shifts sensors into and out of subnets to reach a minimum cost.

*Corresponding Author

The sensor network can be modelled as a set of directed graphs. In this model, the sensors in a subnet are nodes of a graph, and the UAV path through the subnet are directed links between the nodes. Over the years, several approaches have been proposed to find paths through a graph. In the robotics literature, finding a path through a graph has tended to focus on the problem of finding an *open* path through a graph, see for example [1], [2] and [7]. Algorithms for such path planning include Dijkstra's algorithm [3], [5], and the A^* algorithm [6]. However, these algorithms are not applicable to this problem as they find the shortest path between two nodes, not a closed path.

For a single UAV, this path planning problem is the well-known Travelling Salesman Problem (cf. [5], [8] and [9]). Our heuristic approach, using good approximate solutions and making only minor alterations to them, is quite different from conventional techniques which consider much more general distortions of existing paths. For example, we found that our method performs significantly better than synthetic annealing methods, e.g., that found in [10].

2 Adaptive Path Planning Algorithm

Given N sensors and K UAVs, the adaptive path planning algorithm generates K non-overlapping, non-branching, closed paths to every sensor in the network. The sensor network can be modelled as a family of graphs $\{(S_k, P_k)\}$, $k = 1, \ldots, K$ where $S_k = \{s_i\}_k, i = 1, \ldots, N_k$ is the set of sensors in the k^{th} subnet, and $P_k = \{l_{ij}\}$ is the set of weighted, directed links that connect sensor s_i to sensor s_j. Weight d_{ij} associated with link l_{ij} is the distance between sensors s_i and s_j. The length of path P_k, denoted as D_k, is the sum of the weights of the links in P_k.

It is evident that an exhaustive search through all possible paths is $\mathcal{O}(N!)$. In order to lessen the computational burden, a heuristic approach to path planning has been developed. The basis for the path planning algorithm is the cost function

$$C = \sum_{k=1}^{K} (D_k)^a \tag{1}$$

In equation (1), the individual terms in the summation are the costs that each path contributes to the total cost. To illustrate the behavior of path planning with this cost function, consider an incremental change of paths resulting in changes to path lengths δD_k. For small changes, we have

$$\delta C \approx a \sum_k D_k^{a-1} \delta D_k \tag{2}$$

Thus, D_k^{a-1} is the "weight", or cost per unit length, of path k. Any choice of weights which is monotonic increasing in D_k will tend to equalize the path lengths (by penalizing very long paths). Hence, the exponent a provides a parametric means to combine the desires for minimum total path length and roughly equivalent individual lengths. Values in the range of $3 \leq a \leq 6$ have been found to work well in practice.

The adaptive path planning algorithm begins with an initialization procedure, and then adaptively reacts to one of the four basic exceptions. We begin by describing the initialization procedure.

2.1 Path Initialization

The initialization process consists of three steps: 1) K subnets of sensors are formed within a network of N sensors, 2) non-branching, closed paths to each sensor in each subnet are constructed by minimizing the distance between sensors, and 3) paths over all subnets in the network are balanced using cost function (1). The details of each step are described below.

The first step of the initialization process is to construct K subnets of sensors S_k, from the N sensors in the network. The procedure is based on the K-means algorithm with one notable exception. A K-means algorithm begins by randomly selecting K vectors from a set, and using these vectors as the initial centroids of K clusters [4]. Clusters are formed by assigning each vector in the set to the nearest centroid. A new centroid is computed as the average over all vectors in the cluster. This process continues until the K centroids are fixed.

The approach taken here is similar, except that the random initialization of the K-means algorithm has been abandoned in favor of another approach. (The random initialization technique in the K-means algorithm was found to give poor results.) Instead, we initialized the cluster algorithm by finding sensors that are widely separated. Begin by selecting K sensors at random. Form a set U of the positions of these sensors, and find the two position vectors $x_m, x_n \in U$ that are closest to one another. Discard one of these position vectors, and select a new sensor (i.e., position vector) $x_p \notin U$. Continue the process until the minimum sepa-

ration of position vectors in U is maximized. Once the sensors in the network have been grouped into clusters, paths through each cluster are constructed.

Path P_k through the k^{th} subnet is constructed from a circumferential path around S_k which is recursively expanded to include interior sensors using a greedy algorithm. (Recall that a greedy algorithm selects the optimum choice at each step, with no regard beyond a single step.) The circumferential path around S_k is the convex hull of S_k. As the convex hull computation progresses, an ordered list of sensors is produced that when linked form a circumferential path around the subnet [11].

The circumferential path P forms the initial path for the subnet. Denote the set of sensors that form the convex hull of S_k as H. Observe that H partitions the subnet into two groups of sensors: those on P, and those interior to P. Denote the set of sensors in the interior of the subnet as $Q = S_k - H$. Sensors $s_q \in Q$ are added to path P in positions that minimize their contribution to the global cost (1). The differential cost of adding sensor s_q to the path between sensors s_i and s_j is found by breaking link l_{ij} into links l_{iq} and l_{qj}, and is given by

$$\Delta d_{iqj} = d_{iq} + d_{qj} - d_{ij} \tag{3}$$

Sensors in Q are inserted into path P at the position that minimizes (3). The process of adding sensors in Q to the path continues in this manner until all sensors in the subnet have been assigned a position in the path.

After paths through all K subnets have been generated, the paths are balanced using global cost function (1). Path balancing minimizes the global costs of all paths in the sensor network. The differential cost of delinking sensor s_j from sensors s_i and s_k in path p and inserting it into the link between sensors s_m and s_n in path p' is given by (cf. (1))

$$\Delta C = \Delta C_p + \Delta C_{p'} \tag{4}$$

where

$$\Delta C_p = (D_p - \Delta d_{ijk})^a - (D_p)^a \tag{5}$$

and

$$\Delta C_{p'} = (D_{p'} + \Delta d_{mjn})^a - (D_{p'})^a \tag{6}$$

If a particular combination of j, $\{i,k\}$, $\{m,n\}$, and $\{p,p'\}$ yields a $\Delta C < 0$, then the global path cost will decrease if the move is performed. By testing all sensors in all links of all paths in the network, and moving only those sensors that decrease the global path cost, an optimal path (and subnet) configuration is obtained.

The initialization algorithm has been evaluated up to several hundred sensors and sixteen UAVs. The time required to generate a path for one UAV through 100 sensors takes about 1 second on a 450 MHz processor. The computation scales as $\mathcal{O}(N^3)$, but no attempt has been made to improve the computational efficiency of the algorithm. It should be possible, using simple geometric considerations, to reduce the complexity to $\mathcal{O}(N)$ for large N. Swaps of sensors need only be considered among *nearby* paths, not all paths in the network. Examples of paths produced by the initialization algorithm for 124 sensors and up to sixteen UAVs are presented in a later section.

2.2 Path Adaptation

As previously described, one of four exceptions can trigger subnet and path adaptation: 1) a UAV leaves the network, 2) a UAV enters the network, 3) sensors leave the network and 4) sensors enter the network. The subnet and path adaptation schemes are detailed below for each exception.

Exception 1: UAV leaves the network
When a UAV leaves the network, its sensors must be reassigned to other UAVs. Hence, this exception is identical to adding sensors to existing paths. (See Exception 4 for the procedure used to add sensors to the network.)

Exception 2: UAV enters the network
In this exception, the objective is to assign a group of sensors to the new UAV in an equitable manner, while minimizing disruption to other UAVs in the network. The procedure begins by selecting the path with maximum length, and deassigning its sensors from the associated UAV. The deassigned sensors are reformed into two paths, one for the new UAV and one for the deassigned UAV, using the technique described in Section 2.1 (i.e., form two clusters of sensors, compute the convex hulls of the clusters, and insert interior sensors into the paths). Finally, all paths in the network are rebalanced.

Exception 3: Sensors leave the network
The sensors are deleted from the sets of sensors in the subnets, and the paths rebalanced. (It is quite likely that deleting too many sensors at once, before rebalancing, will lead to poor results. This issue has not been investigated.)

Exception 4: Sensors enter the network
Our initial attempt was to add sensors in a greedy fashion, using cost function (1), and then rebalance

the paths. It was found that this approach often leads to obviously suboptimal paths, e.g. paths that cross each other or themselves. Instead, greedy addition of sensors to the nearest path (in a Euclidean sense) was found to work well.

3 Simulation Results

Several simulation examples are provided to illustrate the subnet and path configurations produced by the algorithm. Figure 1 shows a sensor network of 124 randomly distributed sensors. Figures 2 through 4 illustrate the flight paths of one, ten and sixteen UAVs respectively. Note that while the number of sensors per path is not constant in Figures 3 and 4 (the number of sensors per path varies from seven to eighteen for 10 UAVs, and from five to eleven for 16 UAVs), the cost per path for a given network configuration is roughly equal.

Figures 4 through 6 illustrate the adaptation of the sensor network to the departure of one UAV. Figure 4 illustrates a set of paths for a network of 124 sensors (the same network as before) and sixteen UAVs. Figure 5 shows the network immediately after the departure of the tenth UAV. (The subnet for the tenth UAV is seen in Figure 4 in the lower group of paths just right of center.) Paths 4 and 2 each adjust their paths to absorb the sensors in the tenth subnet. In this figure, no other subnets have been affected. Figure 6 illustrates the network after the adaptation has stabilized. Note that subnets surrounding subnets 2 and 4 have absorbed some of the sensors in those subnets to balance the network. Only subnets (1,9) remain as they were before UAV number 10 left the network.

4 Conclusion

An adaptive path planning algorithm for Unmanned Air Vehicles (UAVs) in a land-based sensor network has been presented. After initialization, the algorithm adapts to several exceptions including the addition or loss of a UAV, and the addition or loss of sensors. The adaptation is driven by a global cost function that seeks to minimize the cost associated with a given subnet (and path) configuration. Several examples are presented that illustrate the path configurations produced by the algorithm as well as an example illustrating the adaptation of the algorithm ot the loss of a UAV. These results indicate that good approximate solutions to this variant of the NP-complex Travelling Salesman Problem can be obtained efficiently compared to conventional techniques.

5 Acknowledgements

This research was performed under a grant from the Laboratory Directed Research and Development program, Lawrence Livermore National Laboratory. The authors would like to thank Dave McCallen, Director of the Center for Complex and Distributed Systems, for his support of the project. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] C. Alexopoulos and P. M. Griffin, "Path planning for a mobile robot," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 2, March/April 1992.

[2] D. Z. Chen, R. J. Szczerba, and J. J. Uhran, Jr., "A framed quadtree approach for determining euclidean shortest paths in a 2-D environment," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 5, October 1997.

[3] E. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, Vol. 1, pg. 269–271, 1959.

[4] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.

[5] L. Foulds, *Graph Theory Applications*, Springer-Verlag, 1992.

[6] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on System Science and Cybernetics*, Vol. 4, No. 2, July 1968.

[7] Y. K. Hwang and N. Ahuja, "Gross motion planning—A survey," *ACM Computing Surveys*, Vol. 24, No. 3, September 1992.

[8] S. Lin, "Computer solutions of the travelling salesman problem," *Bell Systems Technical Journal*, Vol. 44, 1965.

[9] E. L. Lawler, et. al., *The Travelling Salesman Problem*, Wiley-Interscience, New York, 1985.

[10] W. H. Press, et. al., *Numerical Recipes in C*, Cambridge University Press, 1988.

[11] J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1993.

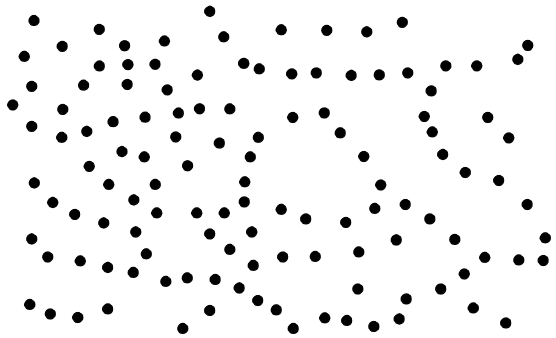

Figure 1: Sensor network of 124 randomly distributed sensors

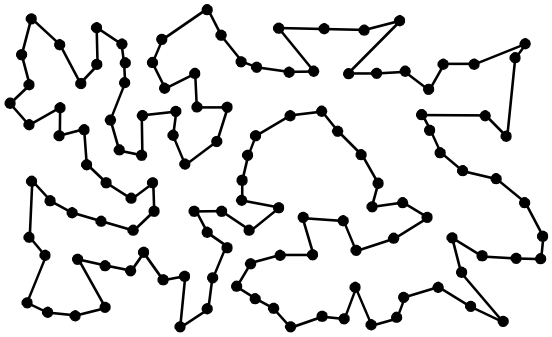

Figure 2: Path through the sensor network for one UAV

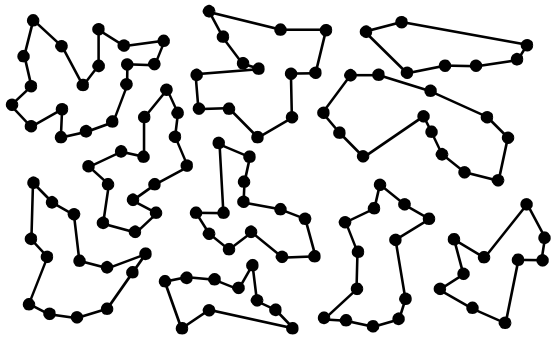

Figure 3: Paths and subnets for ten UAVs

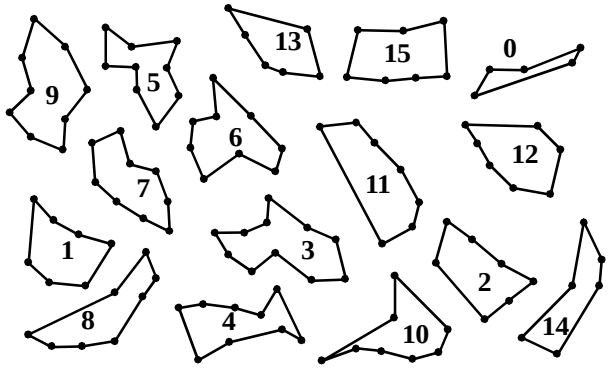

Figure 4: Paths and subnets for sixteen UAVs.

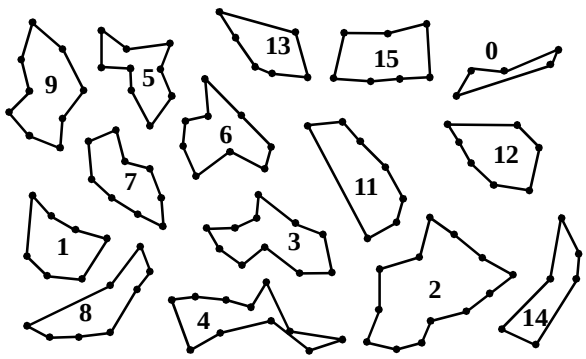

Figure 5: Path rearrangement after the departure of the UAV for subnet number 10.

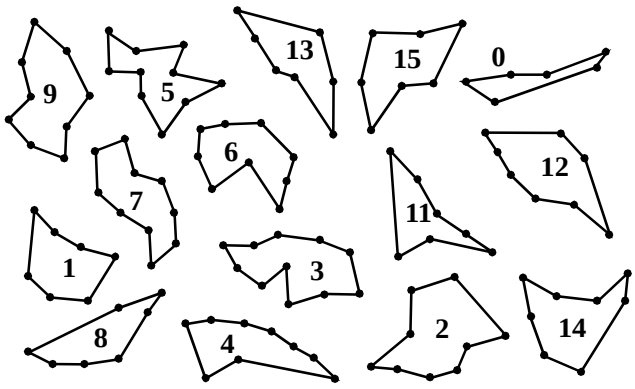

Figure 6: Final path and subnet configuration after the departure of the UAV.